PyPalEx

2.2.0

Generated by Doxygen 1.9.8

1 PyPalEx: The Python Palette Extractor

1.1 Description

PyPalEx is a tool for extracting color palettes from images and generating a JSON format file with light and dark color themes. This tool is intended to be OS independent, for use by the tech community for developing their own custom theme managers or by artists who want to extract color palettes for their art from images, pictures or wallpapers they adore.

2 Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 pypalex Namespace Reference

Python Palette Extractor: extracts color palettes from images.

Namespaces

- namespace __main__
- namespace arg_messages
- namespace constants
- namespace conversion_utils
- namespace extraction_utils
- namespace Extractor
- namespace file_utils
- namespace image_utils
- namespace print_utils

5.1.1 Detailed Description

Python Palette Extractor: extracts color palettes from images.

PyPalEx is a tool for extracting color palettes from images and generating a JSON format file with light and dark color themes. This tool is intended to be OS independent, for use by the tech community for developing their own custom theme managers or by artists who want to extract color palettes for their art from images, pictures or wallpapers they adore.

5.2 pypalex.__main__ Namespace Reference

Functions

- main ()

  Main script function.
- handle_args ()

  Handles the arguments passed to PyPalEx.
- extract_color_palettes ()

  Handles color extraction from image(s).
- setup_argument_parser ()

  Sets up the argument parser for command line arguments.
- check_sources (filepaths, path=None)

Checks each of the sources provided and removes any bad sources.

- check_path (path)

  Check the path to make sure it exists.
- handle_config ()

  Handle the PyPalEx configuration file settings.
- set_global_args (args)

  Sets the global variables using the arguments.
- preview_and_save (extractor, save_type, img_index)

  Shows a preview of and saves the extracted color palette(s).
- check_source (filepath)

  Checks to make sure the path leads to a file.

Variables

- str CONFIG_FILENAME = 'palex-config.yaml'

  Filename of the configuration file.
- list PROPER_IMAGES = [ ]

  List of real/existing image file path(s).
- list FILENAMES = [ ]

  List of image filenames (contain file extensions).
- list IMAGE_NAMES = [ ]

  List of image names.
- str OUTPUT_PATH = ''

  The path to the output directory where all exported files will be saved.
- str EXPORT_FILE_FORMAT = 'json'

  The format of the files to be exported (e.g.
- str EXPORT_COLOR_FORMAT = 'hex'

  The format in which the extracted colors will be exported (e.g.
- dict EXPORT_PALETTE_TEMPLATES = {}

  Dictionary of palette templates that can be used to organize extracted colors into palettes to export.
- dict PALETTE_COLOR_TYPES_CONTAINED = {}

  Dictionary of the color types that are contained within each palette template.
- bool SAVE_CHECK = False

  Flag to check if user wants to save extracted color palettes.
- bool SHOW_PREVIEW = False

  Flag to show a preview of extracted palettes.
- bool ADAPTIVE_PALETTE = False

  Flag to generate 2 adaptive color palettes.
- bool MOOD_PALETTE = False

  Flag to generate 2 mood color palettes.
- bool SAVE_RAW = False

  Flag to save raw extracted colors.
- bool PASTEL_L = False

  Flag to convert light color type to pastel.
- bool PASTEL_N = False

  Flag to convert normal color type to pastel.
- bool PASTEL_D = False

  Flag to convert dark color type to pastel.
- str LIGHT_MOOD_PALETTE_NAME = 'light-mood'

  The palette name of the light-themed mood palette.

- str DARK_MOOD_PALETTE_NAME = 'dark-mood'

  The palette name of the dark-themed mood palette.
- str LIGHT_ADAPTIVE_PALETTE_NAME = 'goldilocks-light'

  The palette name of the light-themed adaptive palette.
- str DARK_ADAPTIVE_PALETTE_NAME = 'goldilocks-dark'

  The palette name of the dark-themed adaptive palette.
- dict VALID_COLOR_SET

  A set of valid color names used to check user-defined color palettes from the configuration file.

5.2.1 Function Documentation

check_path()

```
check_path (
              path )
```

Check the path to make sure it exists.

Parameters

path	The path to a directory.

Returns

True if the path exists and is not a file, False otherwise.

check_source()

```
check_source (
              filepath )
```

Checks to make sure the path leads to a file.

Parameters

filepath	Path to file with filename and file extension.

Returns

True if file exists, False otherwise.

check_sources()

```
check_sources (
              filepaths,
              path = None )
```

Checks each of the sources provided and removes any bad sources.

Any filepaths or source files that are not images or do not exist get removed.

Parameters

filepaths	List of file paths.
path	A path to the images, if it is provided.

Returns

True if all/some sources are good, False if all sources are bad.

extract_color_palettes()

```
extract_color_palettes ( )
```

Handles color extraction from image(s).

handle_args()

```
handle_args ( )
```

Handles the arguments passed to PyPalEx.

handle_config()

```
handle_config ( )
```

Handle the PyPalEx configuration file settings.

main()

```
main ( )
```

Main script function.

preview_and_save()

```
preview_and_save (
              extractor,
              save_type,
              img_index )
```

Shows a preview of and saves the extracted color palette(s).

Parameters

extractor	An Extractor object.
save_type	A string character that specifies what type of palette to extract and save (i.e. a = adaptive, m = mood, r = raw, t = templates).
img_index	The integer index used to identify the image name.

set_global_args()

```
set_global_args (
              args )
```

Sets the global variables using the arguments.

Parameters

args	User-supplied arguments.

setup_argument_parser()

```
setup_argument_parser ( )
```

Sets up the argument parser for command line arguments.

Returns

A command line argument parsing object.

5.2.2 Variable Documentation

ADAPTIVE_PALETTE

```
bool ADAPTIVE_PALETTE = False
```

Flag to generate 2 adaptive color palettes.

CONFIG_FILENAME

```
str CONFIG_FILENAME = 'palex-config.yaml'
```

Filename of the configuration file.

DARK_ADAPTIVE_PALETTE_NAME

```
str DARK_ADAPTIVE_PALETTE_NAME = 'goldilocks-dark'
```

The palette name of the dark-themed adaptive palette.

DARK_MOOD_PALETTE_NAME

```
str DARK_MOOD_PALETTE_NAME = 'dark-mood'
```

The palette name of the dark-themed mood palette.

EXPORT_COLOR_FORMAT

```
str EXPORT_COLOR_FORMAT = 'hex'
```

The format in which the extracted colors will be exported (e.g.

'hsv', 'rgb', 'hex', 'ansi').

EXPORT_FILE_FORMAT

```
str EXPORT_FILE_FORMAT = 'json'
```

The format of the files to be exported (e.g.

'json', 'yaml').

EXPORT_PALETTE_TEMPLATES

```
dict EXPORT_PALETTE_TEMPLATES = {}
```

Dictionary of palette templates that can be used to organize extracted colors into palettes to export.

FILENAMES

```
list FILENAMES = []
```

List of image filenames (contain file extensions).

IMAGE_NAMES

```
list IMAGE_NAMES = []
```

List of image names.

LIGHT_ADAPTIVE_PALETTE_NAME

```
str LIGHT_ADAPTIVE_PALETTE_NAME = 'goldilocks-light'
```

The palette name of the light-themed adaptive palette.

LIGHT_MOOD_PALETTE_NAME

```
str LIGHT_MOOD_PALETTE_NAME = 'light-mood'
```

The palette name of the light-themed mood palette.

MOOD_PALETTE

```
bool MOOD_PALETTE = False
```

Flag to generate 2 mood color palettes.

OUTPUT_PATH

```
str OUTPUT_PATH = ''
```

The path to the output directory where all exported files will be saved.

PALETTE_COLOR_TYPES_CONTAINED

```
dict PALETTE_COLOR_TYPES_CONTAINED = {}
```

Dictionary of the color types that are contained within each palette template.

PASTEL_D

```
bool PASTEL_D = False
```

Flag to convert dark color type to pastel.

PASTEL_L

```
bool PASTEL_L = False
```

Flag to convert light color type to pastel.

PASTEL_N

```
bool PASTEL_N = False
```

Flag to convert normal color type to pastel.

PROPER_IMAGES

```
list PROPER_IMAGES = []
```

List of real/existing image file path(s).

SAVE_CHECK

```
bool SAVE_CHECK = False
```

Flag to check if user wants to save extracted color palettes.

SAVE_RAW

```
bool SAVE_RAW = False
```

Flag to save raw extracted colors.

SHOW_PREVIEW

```
bool SHOW_PREVIEW = False
```

Flag to show a preview of extracted palettes.

VALID_COLOR_SET

```
dict VALID_COLOR_SET
```

Initial value:

```
00001 =  {'red', 'light red', 'dark red', 'orange', 'light orange', 'dark orange',
00002                   'yellow', 'light yellow', 'dark yellow', 'chartreuse', 'light chartreuse', 'dark
      chartreuse',
00003                   'green', 'light green', 'dark green', 'spring', 'light spring', 'dark spring',
00004                   'cyan', 'light cyan', 'dark cyan', 'azure', 'light azure', 'dark azure',
00005                   'blue', 'light blue', 'dark blue', 'violet', 'light violet', 'dark violet',
00006                   'magenta', 'light magenta', 'dark magenta', 'rose', 'light rose', 'dark rose'}
```

A set of valid color names used to check user-defined color palettes from the configuration file.

5.3 pypalex.arg_messages Namespace Reference

Functions

- bad_source_message ()

  Generates an error message if the sources provided were not images.
- bad_path_message ()

  Generates an error message if the directory provided is not a valid directory.
- no_args_help_message ()

  Generates a help message if no arguments were presented.

5.3.1 Function Documentation

bad_path_message()

```
bad_path_message ( )
```

Generates an error message if the directory provided is not a valid directory.

Returns

The "bad directory" message.

bad_source_message()

```
bad_source_message ( )
```

Generates an error message if the sources provided were not images.

Returns

The "bad sources" message.

no_args_help_message()

```
no_args_help_message ( )
```

Generates a help message if no arguments were presented.

Returns

The "no arguments" help message.

5.4 pypalex.constants Namespace Reference

Variables

- list BLACK_RGB = [0, 0, 0]
- list WHITE_RGB = [255, 255, 255]
- list RED_RGB = [255, 0, 0]
- list ORANGE_RGB = [255, 153, 0]
- list YELLOW_RGB = [255, 213, 0]
- list CHARTREUSE_RGB = [191, 255, 0]
- list GREEN_RGB = [0, 255, 0]
- list SPRING_RGB = [0, 255, 149]
- list CYAN_RGB = [0, 255, 255]
- list AZURE_RGB = [0, 128, 255]
- list BLUE_RGB = [0, 0, 255]
- list VIOLET_RGB = [140, 0, 255]
- list MAGENTA_RGB = [255, 0, 255]
- list ROSE_RGB = [255, 0, 93]
- int BLACK_HEX = 0x000000
- int WHITE_HEX = 0xFFFFFF
- int RED_HEX = 0xFF0000
- int ORANGE_HEX = 0xFF9900
- int YELLOW_HEX = 0xFFD500
- int CHARTREUSE_HEX = 0xBFFF00
- int GREEN_HEX = 0x00FF00
- int SPRING_HEX = 0x00FF95
- int CYAN_HEX = 0x00FFFF
- int AZURE_HEX = 0x0080FF
- int BLUE_HEX = 0x0000FF
- int VIOLET_HEX = 0x8C00FF
- int MAGENTA_HEX = 0xFF00FF
- int ROSE_HEX = 0xFF005D

- int RED_HUE = 0
- int ORANGE_HUE = 36
- int YELLOW_HUE = 50
- int CHARTREUSE_HUE = 75
- int GREEN_HUE = 120
- int SPRING_HUE = 155
- int CYAN_HUE = 180
- int AZURE_HUE = 210
- int BLUE_HUE = 240
- int VIOLET_HUE = 273
- int MAGENTA_HUE = 300
- int ROSE_HUE = 338
- list RED_HUE_RANGE_MIN = [0, 20]
- list ORANGE_HUE_RANGE = [20, 43]
- list YELLOW_HUE_RANGE = [43, 64]
- list CHARTREUSE_HUE_RANGE = [64, 90]
- list GREEN_HUE_RANGE = [90, 145]
- list SPRING_HUE_RANGE = [145, 170]
- list CYAN_HUE_RANGE = [170, 195]
- list AZURE_HUE_RANGE = [195, 220]
- list BLUE_HUE_RANGE = [220, 255]
- list VIOLET_HUE_RANGE = [255, 290]
- list MAGENTA_HUE_RANGE = [290, 325]
- list ROSE_HUE_RANGE = [325, 350]
- list RED_HUE_RANGE_MAX = [350, 360]
- list BLACK_BRIGHTNESS_RANGE = [0.0, 35.0]
- list DARK_BRIGHTNESS_RANGE = [35.0, 55.0]
- list NORM_BRIGHTNESS_RANGE = [55.0, 80.0]
- list LIGHT_BRIGHTNESS_RANGE = [80.0, 100.0]
- list SATURATION_TOLERANCE_RANGE = [15.0, 20.0]
- list PASTEL_SATURATION_RANGE = [20.0, 55.0]
- list PASTEL_BRIGHTNESS_RANGE = [65.0, 95.0]

5.4.1 Variable Documentation

AZURE_HEX

```
int AZURE_HEX = 0x0080FF
```

AZURE_HUE

```
int AZURE_HUE = 210
```

AZURE_HUE_RANGE

```
list AZURE_HUE_RANGE = [195, 220]
```

AZURE_RGB

```
list AZURE_RGB = [0, 128, 255]
```

BLACK_BRIGHTNESS_RANGE

```
list BLACK_BRIGHTNESS_RANGE = [0.0, 35.0]
```

BLACK_HEX

```
int BLACK_HEX = 0x000000
```

BLACK_RGB

```
list BLACK_RGB = [0, 0, 0]
```

BLUE_HEX

```
int BLUE_HEX = 0x0000FF
```

BLUE_HUE

```
int BLUE_HUE = 240
```

BLUE_HUE_RANGE

```
list BLUE_HUE_RANGE = [220, 255]
```

BLUE_RGB

```
list BLUE_RGB = [0, 0, 255]
```

CHARTREUSE_HEX

```
int CHARTREUSE_HEX = 0xBFFF00
```

CHARTREUSE_HUE

```
int CHARTREUSE_HUE = 75
```

CHARTREUSE_HUE_RANGE

```
list CHARTREUSE_HUE_RANGE = [64, 90]
```

CHARTREUSE_RGB

```
list CHARTREUSE_RGB = [191, 255, 0]
```

CYAN_HEX

```
int CYAN_HEX = 0x00FFFF
```

CYAN_HUE

```
int CYAN_HUE = 180
```

CYAN_HUE_RANGE

```
list CYAN_HUE_RANGE = [170, 195]
```

CYAN_RGB

```
list CYAN_RGB = [0, 255, 255]
```

DARK_BRIGHTNESS_RANGE

```
list DARK_BRIGHTNESS_RANGE = [35.0, 55.0]
```

GREEN_HEX

```
int GREEN_HEX = 0x00FF00
```

GREEN_HUE

```
int GREEN_HUE = 120
```

GREEN_HUE_RANGE

```
list GREEN_HUE_RANGE = [90, 145]
```

GREEN_RGB

```
list GREEN_RGB = [0, 255, 0]
```

LIGHT_BRIGHTNESS_RANGE

```
list LIGHT_BRIGHTNESS_RANGE = [80.0, 100.0]
```

MAGENTA_HEX

```
int MAGENTA_HEX = 0xFF00FF
```

MAGENTA_HUE

```
int MAGENTA_HUE = 300
```

MAGENTA_HUE_RANGE

```
list MAGENTA_HUE_RANGE = [290, 325]
```

MAGENTA_RGB

```
list MAGENTA_RGB = [255, 0, 255]
```

NORM_BRIGHTNESS_RANGE

```
list NORM_BRIGHTNESS_RANGE = [55.0, 80.0]
```

ORANGE_HEX

```
int ORANGE_HEX = 0xFF9900
```

ORANGE_HUE

```
int ORANGE_HUE = 36
```

ORANGE_HUE_RANGE

```
list ORANGE_HUE_RANGE = [20, 43]
```

ORANGE_RGB

```
list ORANGE_RGB = [255, 153, 0]
```

PASTEL_BRIGHTNESS_RANGE

```
list PASTEL_BRIGHTNESS_RANGE = [65.0, 95.0]
```

PASTEL_SATURATION_RANGE

```
list PASTEL_SATURATION_RANGE = [20.0, 55.0]
```

RED_HEX

```
int RED_HEX = 0xFF0000
```

RED_HUE

```
int RED_HUE = 0
```

RED_HUE_RANGE_MAX

```
list RED_HUE_RANGE_MAX = [350, 360]
```

RED_HUE_RANGE_MIN

```
list RED_HUE_RANGE_MIN = [0, 20]
```

RED_RGB

```
list RED_RGB = [255, 0, 0]
```

ROSE_HEX

```
int ROSE_HEX = 0xFF005D
```

ROSE_HUE

```
int ROSE_HUE = 338
```

ROSE_HUE_RANGE

```
list ROSE_HUE_RANGE = [325, 350]
```

ROSE_RGB

```
list ROSE_RGB = [255, 0, 93]
```

SATURATION_TOLERANCE_RANGE

```
list SATURATION_TOLERANCE_RANGE = [15.0, 20.0]
```

SPRING_HEX

```
int SPRING_HEX = 0x00FF95
```

SPRING_HUE

```
int SPRING_HUE = 155
```

SPRING_HUE_RANGE

```
list SPRING_HUE_RANGE = [145, 170]
```

SPRING_RGB

```
list SPRING_RGB = [0, 255, 149]
```

VIOLET_HEX

```
int VIOLET_HEX = 0x8C00FF
```

VIOLET_HUE

```
int VIOLET_HUE = 273
```

VIOLET_HUE_RANGE

```
list VIOLET_HUE_RANGE = [255, 290]
```

VIOLET_RGB

```
list VIOLET_RGB = [140, 0, 255]
```

WHITE_HEX

```
int WHITE_HEX = 0xFFFFFF
```

WHITE_RGB

```
list WHITE_RGB = [255, 255, 255]
```

YELLOW_HEX

```
int YELLOW_HEX = 0xFFD500
```

YELLOW_HUE

```
int YELLOW_HUE = 50
```

YELLOW_HUE_RANGE

```
list YELLOW_HUE_RANGE = [43, 64]
```

YELLOW_RGB

```
list YELLOW_RGB = [255, 213, 0]
```

5.5 pypalex.conversion_utils Namespace Reference

Functions

- hsv_to_hex (hsv_array)
  Convert HSV array [h,s,v] to HEX string '#ffffff'.
- hex_to_hsv (hex_str)
  Convert HEX string '#ffffff' to HSV array [h,s,v].
- hsv_to_ansi (hsv_array, background=False)
  Convert HSV array [h,s,v] to an ANSI color escape code string.
- ansi_to_hsv (ansi_string)
  Converts ANSI color escape code string tp HSV array [h,s,v].
- hex_to_ansi (hex_str, background=False)
  Convert HEX string '#ffffff' to an ANSI color escape code string.
- ansi_to_hex (ansi_string)
  Converts ANSI color escape code string to HEX string '#ffffff'.
- rgb_to_hsv (rgb_array)
  Converts RGB array [r,g,b] to HSV array [h,s,v].
- hsv_to_rgb (hsv_array)
  Convert HSV array [h,s,v] to RGB array [r,g,b].
- rgb_to_hex (rgb_array)
  Convert RGB array [r,g,b] to HEX string '#ffffff'.
- hex_to_rgb (hex_str)
  Convert HEX string '#ffffff' to RGB array [r,g,b].
- rgb_to_ansi (rgb_array, background=False)
  Convert RGB array [r,g,b] to an ANSI color escape code string.
- ansi_to_rgb (ansi_string)
  Converts ANSI color escape code string to an RGB array.

5.5.1 Function Documentation

ansi_to_hex()

```
ansi_to_hex (
              ansi_string )
```

Converts ANSI color escape code string to HEX string '#ffffff'.

ANSI where \033[38;2;r;g;bm is for the foreground color and \033[48;2;r;g;bm is for the background color. HEX string is in the set ["#000000", "#ffffff"].

Parameters

ansi_string	ANSI color escape code string.

Returns

A HEX string.

ansi_to_hsv()

```
ansi_to_hsv (
              ansi_string )
```

Converts ANSI color escape code string tp HSV array [h,s,v].

ANSI where \033[38;2;r;g;bm is for the foreground color and \033[48;2;r;g;bm is for the background color. HSV where h is in the set [0, 359] and s, v are in the set [0.0, 100.0].

Parameters

ansi_string	ANSI color escape code string.

Returns

HSV array [h,s,v].

ansi_to_rgb()

```
ansi_to_rgb (
              ansi_string )
```

Converts ANSI color escape code string to an RGB array.

Note

This function is dependent on the ANSI string to be formatted like '\033[{};2;{};{};{}m' or '\u001b[{};2;{};{};{}m' or something similar. For more information about these ANSI escape codes, here are some sources: https://en.wikipedia.org/wiki/ANSI_escape_code#8-bit https↩
://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences/33206814#
https://stackoverflow.com/questions/45782766/color-python-output-given-rrggbb-hex-v

Parameters

ansi_string	ANSI color escape code string.

Returns

RGB array [r,g,b].

hex_to_ansi()

```
hex_to_ansi (
              hex_str,
              background = False )
```

Convert HEX string '#ffffff' to an ANSI color escape code string.

HEX string is in the set ["#000000", "#ffffff"]. ANSI where \033[38;2;r;g;bm is for the foreground color and \033[48;2;r;g;bm is for the background color.

Parameters

hex_str	HEX string '#ffffff'.
background	Flag for if the HEX string is for a background or not.

Returns

ANSI escape code string.

hex_to_hsv()

```
hex_to_hsv (
              hex_str )
```

Convert HEX string '#ffffff' to HSV array [h,s,v].

HEX string is in the set ["#000000", "#ffffff"]. HSV where h is in the set [0, 359] and s, v are in the set [0.0, 100.0].

Parameters

hex_str	HEX string '#ffffff'.

Returns

HSV array [h,s,v].

hex_to_rgb()

```
hex_to_rgb (
```

```
            hex_str )
```

Convert HEX string '#ffffff' to RGB array [r,g,b].

HEX string is in the set ["#000000", "#ffffff"]. RGB where [r,g,b] are in the set [0, 255].

Parameters

hex_str	HEX string '#ffffff'.

Returns

RGB array [r,g,b].

hsv_to_ansi()

```
hsv_to_ansi (
            hsv_array,
            background = False )
```

Convert HSV array [h,s,v] to an ANSI color escape code string.

HSV where h is in the set [0, 359] and s, v are in the set [0.0, 100.0]. ANSI where \033[38;2;r;g;bm is for the foreground color and \033[48;2;r;g;bm is for the background color.

Parameters

hsv_array	HSV array [h,s,v].
background	Flag for if the HSV color is for a background or not.

Returns

ANSI escape code string.

hsv_to_hex()

```
hsv_to_hex (
            hsv_array )
```

Convert HSV array [h,s,v] to HEX string '#ffffff'.

HSV where h is in the set [0, 359] and s, v are in the set [0.0, 100.0]. HEX string is in the set ["#000000", "#ffffff"].

Parameters

hsv_array	HSV array [h,s,v].

Returns

A HEX string.

hsv_to_rgb()

```
hsv_to_rgb (
              hsv_array )
```

Convert HSV array [h,s,v] to RGB array [r,g,b].

HSV where h is in the set [0, 359] and s, v are in the set [0.0, 100.0]. RGB where [r,g,b] are in the set [0, 255]. Formula adapted from https://www.rapidtables.com/convert/color/hsv-to-rgb.html

Parameters

hsv_array	HSV array [h,s,v].

Returns

RGB array [r,g,b].

rgb_to_ansi()

```
rgb_to_ansi (
              rgb_array,
              background = False )
```

Convert RGB array [r,g,b] to an ANSI color escape code string.

An RGB [r,g,b] array is used to generate an ANSI escape code of the RGB color for use in the terminal CLI. The basic format for these codes depends on if it will be used for foreground or background color. Use \033[38;2;r;g;bm for the foreground color. Use \033[48;2;r;g;bm for the background color.

Note

For more information about these ANSI escape codes, here are some sources: https://en.wikipedia.org/wiki/ANSI_escape_code#8-bit https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences/33206814#33206814 https://stackoverflow.com/questions/45782766/color-python-output-given-rrggbb-hex-v

Parameters

rgb_array	RGB array [r,g,b].
background	Flag for if the RGB color is for a background or not.

Returns

ANSI escape code string of the RGB color.

rgb_to_hex()

```
rgb_to_hex (
              rgb_array )
```

Convert RGB array [r,g,b] to HEX string '#ffffff'.

RGB where [r,g,b] are in the set [0, 255]. HEX string is in the set ["#000000", "#ffffff"].

Parameters

rgb_array	RGB array [r,g,b].

Returns

A HEX string.

rgb_to_hsv()

```
rgb_to_hsv (
              rgb_array )
```

Converts RGB array [r,g,b] to HSV array [h,s,v].

RGB where [r,g,b] are in the set [0, 255]. HSV where h is in the set [0, 359] and s, v are in the set [0.0, 100.0]. Formula adapted from https://www.rapidtables.com/convert/color/rgb-to-hsv.html

Parameters

rgb_array	RGB array [r,g,b].

Returns

HSV array [h,s,v].

5.6 pypalex.extraction_utils Namespace Reference

Functions

- extract_ratios (hsv_img_matrix_2d)

  Extracts the ratios of hues per pixel.
- construct_base_color_dictionary (hsv_img_matrix_2d)

  Constructs dictionary of base colors from an array of HSV pixel values.
- extract_colors (base_color_dict, ratios=None)

  Extracts dominant light, normal and dark colors from each of the base colors.
- check_missing_colors (base_color_dict, extracted_colors_dict)

  Checks for any missing colors in the base color dictionary and borrows them from the surrounding colors.
- generate_remaining_colors (extracted_colors_dict, ratios)

  Generate the remaining black and white, and background and foreground colors.

- extract_color_types (color_data)

  Extracts the dominant color types from a base color.
- set_missing_color (extracted_colors_dict, color_name)

  Sets a specified color in the extracted colors dictionary by borrowing from other extracted colors.
- get_dominant_hue (extracted_colors_dict, ratios)

  Calculates the dominant hue.
- generate_black_and_white (dominant_hue)

  Generates black and white color types using the dominant hue.
- generate_background_and_foreground (dominant_hue, complementary_hue)

  Generates the background and foreground colors.
- sort_by_sat_and_bright_value (hsv_base_color_matrix, color_name, ratios)

  Sorts the colors by their saturation and brightness values.
- extract_dominant_color (hsv_color_type_matrix)

  Extracts the dominant color from a color type.
- check_missing_color_types (light_color, norm_color, dark_color, black_color, achromatic_light, achromatic_norm, achromatic_dark, achromatic_black)

  Checks to make sure all the color types have been properly set.
- get_left_and_right_colors (origin_color_name, color_type='')

  Gets the color names of the colors that are to the left and right of the originating color.
- borrow_color (extracted_colors_dict, origin, borrow_left, borrow_right, color_type='')

  Borrows a color from one of the extracted color types of the base colors.
- get_dominant_color_name (ratios)

  Get the base name of the dominant color from a dictionary of ratios.
- calculate_centroid (hsv_color_type_matrix)

  Calculates the centroid for a color type.
- find_closest_to_centroid (hsv_color_type_matrix, centroid)

  Finds a color from a color type that is closest to the centroid.
- get_hue_shift_value (hue, shift_percentage)

  Gets the appropriate percentage to shift a hue value based on the hue provided.
- calculate_dist_between_2_colors (hsv_color1, hsv_color2)

  Calculates the distance between 2 HSV colors.

5.6.1 Function Documentation

borrow_color()

```
borrow_color (
              extracted_colors_dict,
              origin,
              borrow_left,
              borrow_right,
              color_type = '' )
```

Borrows a color from one of the extracted color types of the base colors.

Parameters

extracted_colors_dict	A dictionary of extracted colors.
origin	A string that represents the name of the originating color.
borrow_left	A string that represents the name of the color to borrow from, to the left of origin.
borrow_right	A string that represents the name of the color to borrow from, to the right of origin.
color_type	A string that represents the type of color ('light', 'dark', '' for normal).

Returns

A numpy array of a borrowed color.

calculate_centroid()

```
calculate_centroid (
              hsv_color_type_matrix )
```

Calculates the centroid for a color type.

The centroid is basically the average color of a set of colors in [h,s,v] format. The centroid is a point in 3-dimensional space. The following sources were used to make this algorithm: http://mkweb.bcgsc.ca/color-summarizer/?faq#averagehue and https://stackoverflow.com/a/8170595/17047816

Parameters

hsv_color_type_matrix	A 2D numpy array of a color type in [h,s,v] format.

Returns

List of centroid color values in [h,s,l] format.

calculate_dist_between_2_colors()

```
calculate_dist_between_2_colors (
              hsv_color1,
              hsv_color2 )
```

Calculates the distance between 2 HSV colors.

Parameters

hsv_color1	A list or numpy array of a color in HSV format [h, s, v].
hsv_color2	A list or numpy array of a color in HSV format [h, s, v].

Returns

A float value that represents the distance between 2 colors.

check_missing_color_types()

```
check_missing_color_types (
              light_color,
              norm_color,
              dark_color,
              black_color,
              achromatic_light,
```

```
                achromatic_norm,
                achromatic_dark,
                achromatic_black )
```

Checks to make sure all the color types have been properly set.

If a color type is missing, then it will be derived from the existing color types.

Note

I'm using the normalization formula from https://stats.stackexchange.com/a/281164

Parameters

light_color	A numpy array of a light color type in [h,s,v] format.
norm_color	A numpy array of a normal color type in [h,s,v] format.
dark_color	A numpy array of a dark color type in [h,s,v] format.
black_color	A numpy array of a black color type in [h,s,v] format.
achromatic_light	A numpy array of an achromatic light color type in [h,s,v] format.
achromatic_norm	A numpy array of an achromatic normal color type in [h,s,v] format.
achromatic_dark	A numpy array of an achromatic dark color type in [h,s,v] format.
achromatic_black	A numpy array of an achromatic black color type in [h,s,v] format.

check_missing_colors()

```
check_missing_colors (
                base_color_dict,
                extracted_colors_dict )
```

Checks for any missing colors in the base color dictionary and borrows them from the surrounding colors.

Parameters

base_color_dict	A dictionary of 2D numpy arrays for each of the base colors.
extracted_colors_dict	A dictionary of extracted colors.

construct_base_color_dictionary()

```
construct_base_color_dictionary (
                hsv_img_matrix_2d )
```

Constructs dictionary of base colors from an array of HSV pixel values.

Base colors are classified as [red, orange, yellow, chartreuse, green, spring, cyan, azure, blue, violet, magenta, rose].

Note

This function operates on the assumption that all the elements in the hsv_img_matrix_2d array are sorted in ascending order using the hue (h) value.

Parameters

hsv_img_matrix_2d	A 2D numpy array of pixels from an image, in [h,s,v] format.

Returns

Dictionary of base colors.

extract_color_types()

```
extract_color_types (
              color_data )
```

Extracts the dominant color types from a base color.

A color type is either a light, normal or dark version of a base color.

Parameters

color_data	A tuple whose elements are a 2D numpy array of a base color, color name string and ratios dictionary.

Returns

List of dominant color types, where each color type is a numpy array in [h,s,v] format.

extract_colors()

```
extract_colors (
              base_color_dict,
              ratios = None )
```

Extracts dominant light, normal and dark colors from each of the base colors.

Parameters

base_color_dict	A dictionary of 2D numpy arrays for each of the base colors.
ratios	A dictionary of color ratios (percentages) in set [0.0, 100.0] for each of the base colors.

Returns

Dictionary of light, normal and dark color types for each of the base colors.

extract_dominant_color()

```
extract_dominant_color (
              hsv_color_type_matrix )
```

Extracts the dominant color from a color type.

A color type is either a light, normal, or dark version of a base color.

Parameters

hsv_color_type_matrix	A 2D numpy array of a color type where every element is a list in [h,s,v] format.

Returns

A numpy array of a dominant color in [h,s,v] format.

extract_ratios()

```
extract_ratios (
              hsv_img_matrix_2d )
```

Extracts the ratios of hues per pixel.

Parameters

hsv_img_matrix_2d	A 2D numpy array of pixels, where each element/pixel is a list of color values in [h,s,v] format.

Returns

Dictionary of hue ratios (percentage) in set [0.0, 100.0]

find_closest_to_centroid()

```
find_closest_to_centroid (
              hsv_color_type_matrix,
              centroid )
```

Finds a color from a color type that is closest to the centroid.

The distance between the centroid color and each of the other individual colors is calculated in 3-dimensional space using the Euclidean Distance formula from the following sources: https://stackoverflow.com/a/35114586/17047816 and https://byjus.com/maths/distance-between-two-points-3d/.

Parameters

hsv_color_type_matrix	A 2D numpy array of a color type where every element is a list in [h,s,v] format.
centroid	List of centroid color values in [h,s,l] format.

Returns

List of all the colors in [h,s,v] format that are the shortest distance away from the centroid.

generate_background_and_foreground()

```
generate_background_and_foreground (
              dominant_hue,
              complementary_hue )
```

Generates the background and foreground colors.

The background and foreground colors are based on the dominant hue in an image and it's complimentary hue. The saturation and brightness values for the background and foreground colors need to be hardcoded to be easier to look at.

Parameters

dominant_hue	An integer that represents the dominant hue in an image.
complementary_hue	An integer that represents the complimentary hue to the dominant hue.

Returns

A list of light and dark background and foreground colors in [h,s,v] format.

generate_black_and_white()

```
generate_black_and_white (
              dominant_hue )
```

Generates black and white color types using the dominant hue.

The saturation and brightness values, for the black and white color types, needs to be hardcoded in order to not interfere with the background and foreground colors.

Parameters

dominant_hue	An integer that represents the dominant hue in an image.

Returns

List of black and white color types in [h,s,v] format.

generate_remaining_colors()

```
generate_remaining_colors (
              extracted_colors_dict,
              ratios )
```

Generate the remaining black and white, and background and foreground colors.

Parameters

extracted_colors_dict	A Dictionary of extracted colors.
ratios	A Dictionary of ratios of the base colors in the image.

get_dominant_color_name()

```
get_dominant_color_name (
              ratios )
```

Get the base name of the dominant color from a dictionary of ratios.

Parameters

ratios	A dictionary that contains ratios for the 12 base colors as well as for each of their color types.

Returns

A string that represents the name of the dominant color from one of the 12 base colors.

get_dominant_hue()

```
get_dominant_hue (
              extracted_colors_dict,
              ratios )
```

Calculates the dominant hue.

The dominant hue, also referred to as the average hue, is based on the color ratios and the colors extracted from an image.

Parameters

extracted_colors_dict	A Dictionary of extracted colors.
ratios	A Dictionary of ratios of the colors in the image (contains base and type color ratios).

Returns

An integer that represents the dominant hue in an image.

get_hue_shift_value()

```
get_hue_shift_value (
              hue,
              shift_percentage )
```

Gets the appropriate percentage to shift a hue value based on the hue provided.

The shift percentage tells use by how much to shift a hue value based on what range the hue is in.

Parameters

hue	An integer that represents a hue value in range [0, 359].
shift_percentage	A float that represents the percentage in range [0.0, 100.0] by which to shift hue.

Returns

An integer that represents the value by which to shift the hue.

get_left_and_right_colors()

```
get_left_and_right_colors (
              origin_color_name,
              color_type = '' )
```

Gets the color names of the colors that are to the left and right of the originating color.

There are two ways to think about left and right on a color wheel: from the top of the color wheel looking at the top-most color or from the bottom of the color wheel looking at the bottom-most color. This has an effect on how we think of the linear format of the color wheel. For this package we will think about left and right colors using the former option (top of the color wheel).

Note

There are 12 base colors to choose from: red, orange, yellow, chartreuse, green, spring, cyan, azure, blue, violet, magenta and rose.

Parameters

origin_color_name	A string that represents the name of the originating color.
color_type	A string that represents the type of color ('light', 'dark', '' for normal).

Returns

List of color names that are to the left and right of the originating color.

set_missing_color()

```
set_missing_color (
              extracted_colors_dict,
              color_name )
```

Sets a specified color in the extracted colors dictionary by borrowing from other extracted colors.

Note

There are 12 base colors to choose from: red, orange, yellow, chartreuse, green, spring, cyan, azure, blue, violet, magenta and rose.

Parameters

extracted_colors_dict	A dictionary of extracted colors, where each color is a list in HSV format.
color_name	A string that represents one of the 12 base colors which will be set.

sort_by_sat_and_bright_value()

```
sort_by_sat_and_bright_value (
              hsv_base_color_matrix,
              color_name,
              ratios )
```

Sorts the colors by their saturation and brightness values.

A color type is either a light, normal, dark, black or achromatic version of a base color. The ratios for the light, normal and dark color types are also calculated.

Parameters

hsv_base_color_matrix	A 2D numpy array of a base color, where each element is a list in [h,s,v] format.
color_name	A string that represents the name of a base color.
ratios	A dictionary of color ratios (percentages) in set [0.0, 100.0] for each of the base colors.

Returns

A list of color types, where each element is a 2D numpy array of a color type whose elements are a list in [h,s,v] format.

5.7 pypalex.Extractor Namespace Reference

Classes

- class Extractor

  Extracts colors given a matrix of HSV values extracted from an image.

5.8 pypalex.file_utils Namespace Reference

Functions

- generate_config_file (config_filename)

  Generates a configuration file.
- raw_dump (extracted_colors_dict, image_name, output_path, export_file_format, export_color_format)

  Saves the raw extracted colors into a file.
- save_palettes (palettes, image_name, output_path, export_file_format, export_color_format, palette_color↩ _types=None)

  Saves the color palettes of extracted colors.

5.8.1 Function Documentation

generate_config_file()

```
generate_config_file (
              config_filename )
```

Generates a configuration file.

Generates a configuration file and saves it in the default Configuration Directory for PyPalEx.

Note

If a file with the same name already exists, it can be overwritten.

Parameters

config_filename	A string that represents the filename of the configuration file (e.g. 'palex-config.yaml').

raw_dump()

```
raw_dump (
              extracted_colors_dict,
              image_name,
              output_path,
              export_file_format,
              export_color_format )
```

Saves the raw extracted colors into a file.

Note

If a file with the same name already exists, it can be overwritten.

Parameters

extracted_colors_dict	A dictionary of colors.
image_name	A string that represents the name of the image from where the colors were extracted (e.g. 'forest_wallpaper', 'bubblegum', etc).
output_path	A string that specifies the directory where to save the file (can be a blank string).
export_file_format	A string that specifies the format of the file that will be exported (e.g. 'json', 'yaml').
export_color_format	A string that specifies the format of the colors that will be exported (e.g. 'hsv', 'rgb', 'hex', 'ansi').

save_palettes()

```
save_palettes (
              palettes,
              image_name,
              output_path,
              export_file_format,
              export_color_format,
              palette_color_types = None )
```

Saves the color palettes of extracted colors.

Each palette is saved to its own individual file.

Note

If files with the same name already exist, they can be overwritten.

Parameters

palettes	A dictionary of palettes that were organized based on the palette templates.
image_name	A string that represents the name of the image from where the colors were extracted (e.g. 'forest_wallpaper', 'bubblegum', etc).
output_path	A string that specifies the directory where to save the file (can be a blank string).
export_file_format	A string that specifies the format of the file that will be exported (e.g. 'json', 'yaml').
export_color_format	A string that specifies the format of the colors that will be exported (e.g. 'hsv', 'rgb', 'hex', 'ansi').
palette_color_types	A dictionary that holds flags (True / False) for the color types contained in each palette and if those color types are pastel or not.

5.9 pypalex.image_utils Namespace Reference

Functions

- process_image (image)

  Processes PIL Image object.
- rescale_image (image)

  Rescales image to a smaller sampling size while maintaining aspect ration.
- process_helper (rgb_matrix_2d)

  Helper function for multiprocessing conversion operations.

5.9.1 Function Documentation

process_helper()

```
process_helper (
              rgb_matrix_2d )
```

Helper function for multiprocessing conversion operations.

Helps convert from [r,g,b] to [h,s,v].

Parameters

rgb_matrix_2d	A 2D matrix of rgb values.

Returns

A numpy array/2D matrix of converted [h,s,v] values.

process_image()

```
process_image (
              image )
```

Processes PIL Image object.

Multiprocessing example from: https://stackoverflow.com/a/45555516

Parameters

image	PIL Image object.

Returns

2D numpy array of [h,s,v] arrays (pixels) from image.

rescale_image()

```
rescale_image (
              image )
```

Rescales image to a smaller sampling size while maintaining aspect ration.

Note

The math behind rescaling the image came from: https://math.stackexchange.com/a/3078131

Parameters

image	PIL Image object.

Returns

Tuple of the new width and height of image.

5.10 pypalex.print_utils Namespace Reference

Functions

- print_raw_colors (extracted_colors_dict, color_format, pair_colors=False)
  Prints raw extracted colors to the Terminal.
- print_palette_preview (palettes, color_format)
  Prints the extracted colors, organized into color palettes, to the Terminal.
- print_default_palette_preview (extracted_colors_dict, color_format)
  Prints the extracted colors, organized into default color palettes, to the Terminal.
- print_template_palette_preview (extracted_colors_dict, palette_templates, color_format)
  Prints the extracted colors, organized with palette templates, to the Terminal.
- print_raw_color (color_name, extracted_color, rgb_color, ansi_color)
  Prints a raw color to the Terminal screen.
- get_rgb_colors (extracted_colors_dict, color_format)
  Constructs a dictionary of colors in RGB [r,g,b] format.
- get_ansi_color_codes (rgb_colors_dict)
  Constructs an ANSI escape code dictionary using a dictionary of colors in RGB [r,g,b] format.
- make_default_row (rgb_row_color, blank_row, border_type=None)
  Creates a string that represents a default row when printing palette previews.

- make_foreground_row (rbg_foreground_color, rbg_background_color)

  Creates a string that represents the foreground row when printing palette previews.
- make_panes (background_ansi_color, standard_ansi_colors, intense_ansi_colors)

  Creates a string that represents the 4 rows of panes when printing palette previews.
- make_panes_row (background_ansi_color, standard_ansi_colors, intense_ansi_colors, panes_section)

  Creates a string that represents a row of panes for printing palette previews.

5.10.1 Function Documentation

get_ansi_color_codes()

```
get_ansi_color_codes (
              rgb_colors_dict )
```

Constructs an ANSI escape code dictionary using a dictionary of colors in RGB [r,g,b] format.

Parameters

rgb_colors_dict	A dictionary of colors in RGB [r,g,b] format.

Returns

A dictionary of ANSI color escape codes.

get_rgb_colors()

```
get_rgb_colors (
              extracted_colors_dict,
              color_format )
```

Constructs a dictionary of colors in RGB [r,g,b] format.

Parameters

extracted_colors_dict	A dictionary of colors.
color_format	A string that specifies the format of each color in the extracted colors dictionary (e.g. 'hsv', 'rgb', 'hex', 'ansi').

Returns

A dictionary of RGB colors.

make_default_row()

```
make_default_row (
              rgb_row_color,
```

```
                blank_row,
                border_type = None )
```

Creates a string that represents a default row when printing palette previews.

The default row can be either a blank row or a row with a specific border.

Parameters

rgb_row_color	The color of the row in RGB [r,g,b] format.
blank_row	Flag that determines if this is a blank row or a border row.
border_type	A string that specifies if this row needs a border (e.g. 'top', 'bottom').

Returns

A string that represents a default row that can be printed.

make_foreground_row()

```
make_foreground_row (
                rbg_foreground_color,
                rbg_background_color )
```

Creates a string that represents the foreground row when printing palette previews.

Parameters

rbg_foreground_color	The foreground color of the row in RGB [r,g,b] format.
rbg_background_color	The background color of the row in RGB [r,g,b] format.

Returns

A string that represents a foreground row that can be printed.

make_panes()

```
make_panes (
                background_ansi_color,
                standard_ansi_colors,
                intense_ansi_colors )
```

Creates a string that represents the 4 rows of panes when printing palette previews.

Parameters

background_ansi_color	An ANSI escape code string of the background color.
standard_ansi_colors	A list of ANSI escape code strings for standard colors.
intense_ansi_colors	A list of ANSI escape code strings for intense colors.

Returns

A string that represents the 4 rows of panes that can be printed.

make_panes_row()

```
make_panes_row (
              background_ansi_color,
              standard_ansi_colors,
              intense_ansi_colors,
              panes_section )
```

Creates a string that represents a row of panes for printing palette previews.

Parameters

background_ansi_color	An ANSI escape code string of the background color.
standard_ansi_colors	A list of ANSI escape code strings for standard colors.
intense_ansi_colors	A list of ANSI escape code strings for intense colors.
panes_section	A string that specifies which section of the panes to make (e.g. 'top', 'middle', 'bottom').

Returns

A string that represents a row of panes that can be printed.

print_default_palette_preview()

```
print_default_palette_preview (
              extracted_colors_dict,
              color_format )
```

Prints the extracted colors, organized into default color palettes, to the Terminal.

Prints a preview of the extracted colors to the user's CLI / Terminal screen, organized into default palettes and using ANSI escape codes and ASCII characters.

Note

The CLI / Terminal needs to be able to display ASCII characters and ANSI colors for this to work.

DEPRECATED function, in favor of print_palette_preview().

Parameters

extracted_colors_dict	A dictionary of colors.
color_format	A string that specifies the format of each color in the extracted colors dictionary (e.g. 'hsv', 'rgb', 'hex', 'ansi').

print_palette_preview()

```
print_palette_preview (
              palettes,
              color_format )
```

Prints the extracted colors, organized into color palettes, to the Terminal.

Prints a preview of the extracted colors to the user's CLI / Terminal screen, organized into palettes and using ANSI escape codes and ASCII characters.

Note

The CLI / Terminal needs to be able to display ASCII characters and ANSI colors for this to work.

Parameters

palettes	A dictionary of color palettes, where each item (palette) is a dictionary of colors.
color_format	A string that specifies the format of each color in the extracted colors dictionary (e.g. 'hsv', 'rgb', 'hex', 'ansi').

print_raw_color()

```
print_raw_color (
              color_name,
              extracted_color,
              rgb_color,
              ansi_color )
```

Prints a raw color to the Terminal screen.

Parameters

color_name	A string name of the color to be printed (i.e. red, yellow, blue, etc.).
extracted_color	The extracted color in any color format.
rgb_color	The extracted color in RGB color format.
ansi_color	The extracted color in ANSI color format.

print_raw_colors()

```
print_raw_colors (
              extracted_colors_dict,
              color_format,
              pair_colors = False )
```

Prints raw extracted colors to the Terminal.

Parameters

extracted_colors_dict	A dictionary of colors.

Parameters

color_format	A string that specifies the format of each color in the extracted colors dictionary (e.g. 'hsv', 'rgb', 'hex', 'ansi').
pair_colors	A Flag to specify whether to print colors by pairing them by color type or not.

print_template_palette_preview()

```
print_template_palette_preview (
              extracted_colors_dict,
              palette_templates,
              color_format )
```

Prints the extracted colors, organized with palette templates, to the Terminal.

Prints a preview of the extracted colors to the user's CLI / Terminal screen, organized with palette templates and using ANSI escape codes and ASCII characters.

Note

The CLI / Terminal needs to be able to display ASCII characters and ANSI colors for this to work.

DEPRECATED function, in favor of print_palette_preview().

Parameters

extracted_colors_dict	A dictionary of colors.
palette_templates	A dictionary of palette templates.
color_format	A string that specifies the format of each color in the extracted colors dictionary (e.g. 'hsv', 'rgb', 'hex', 'ansi').

6 Class Documentation

6.1 Extractor Class Reference

Extracts colors given a matrix of HSV values extracted from an image.

Public Member Functions

- __init__ (self)

  Extractor Constructor.
- load (self, absolute_image_path, image_name=None)

  Loads the Extrator class with the provided image.
- run (self)

  Main method for Extractor class.
- convert_to_pastel (self, pastel_light=False, pastel_normal=False, pastel_dark=False)

  Converts the selected color types from the extracted colors to pastel.

- set_color_format (self, new_color_format, colors_dict=None)

  Sets the color format of the colors in the extracted dictionary.
- generate_palettes (self, palette_templates=None)

  Generates palettes based on a dictionary of palette templates.
- generate_adaptive_palettes (self, light_palette_name='goldilocks-light', dark_palette_name='goldilocks-dark')

  Generates two adaptive palettes based on the dominant color of each color-pair.
- generate_mood_palettes (self, light_palette_name='light-mood', dark_palette_name='dark-mood')

  Generates two mood palettes, starting with the dominant color in the image.
- organize_extracted_dictionary (self)

  Organizes the extracted colors dictionary.
- convert_pastel_light (self)

  Converts light color type to pastel.
- convert_pastel_normal (self)

  Converts normal color type to pastel.
- convert_pastel_dark (self)

  Converts dark color type to pastel.
- generate_default_palettes (self)

  Generates a default set of palettes from the extracted colors.
- get_goldilocks_colorset (self, color_name1, color_name2, dom_color_name)

  Retrieves a list of 3 color types.
- get_color_index (self, color_name)

  Gets the index of a base color as it would appear in a linear cyclical array.
- get_color_code_index (self, color_name)

  Gets the color code index of a base color as it would appear in a color palette.
- convert_pastel (self, hsv_color)

  Converts/normalizes HSV color to pastel.
- get_closest_to_goldilocks (self, color_name1, color_name2, dom_color_name, color_type='')

Public Attributes

- hsv_img_matrix_2d

  A 2D numpy array of pixels from an image in [h,s,v] format.
- image_name

  The name of the image file, without any extension (e.g.
- color_format

  A string that represents the format each color should have (e.g.
- ratio_dict

  A dictionary that holds the ratio of base colors in an image and is used to identify the dominant color in an image.
- base_color_dict

  A dictionary of 2D numpy arrays for each of the 6 base colors.
- extracted_colors_dict

  A dictionary of extracted colors in [h,s,v] format.

6.1.1 Detailed Description

Extracts colors given a matrix of HSV values extracted from an image.

6.1.2 Constructor & Destructor Documentation

__init__()

```
__init__ (
            self )
```

Extractor Constructor.

Parameters

self	The object pointer.

6.1.3 Member Function Documentation

convert_pastel()

```
convert_pastel (
            self,
            hsv_color )
```

Converts/normalizes HSV color to pastel.

For values x in range [a, b], values x can be normalized to the new range [y, z] with the following equation: new_x = y + (((x-a) / (b-a)) ∗ (z-y))

Note

I'm using the normalization formula from https://stats.stackexchange.com/a/281164

Parameters

self	The object pointer.
hsv_color	List HSV color to be converted to pastel.

convert_pastel_dark()

```
convert_pastel_dark (
            self )
```

Converts dark color type to pastel.

Parameters

self	The object pointer.

convert_pastel_light()

```
convert_pastel_light (
              self )
```

Converts light color type to pastel.

Parameters

self	The object pointer.

convert_pastel_normal()

```
convert_pastel_normal (
              self )
```

Converts normal color type to pastel.

Parameters

self	The object pointer.

convert_to_pastel()

```
convert_to_pastel (
              self,
              pastel_light = False,
              pastel_normal = False,
              pastel_dark = False )
```

Converts the selected color types from the extracted colors to pastel.

There are only 3 color types to choose from: light, normal, dark.

Parameters

self	The object pointer.
pastel_light	Flag to convert light color types to pastel.
pastel_normal	Flag to convert normal color types to pastel.
pastel_dark	Flag to convert dark color types to pastel.

generate_adaptive_palettes()

```
generate_adaptive_palettes (
              self,
              light_palette_name = 'goldilocks-light',
              dark_palette_name = 'goldilocks-dark' )
```

Generates two adaptive palettes based on the dominant color of each color-pair.

Color-pairs are pairs of colors that are adjacent on the color wheel. Color-pairs include [rose, red], [orange, yellow], [chartreuse, green], [spring, cyan], [azure, blue] and [violet, magenta].

Note

The Goldilocks naming convention is a tribute to the goldilocks zone (meaning "ideal" colors).

Parameters

self	The object pointer.
light_palette_name	A string representation of the light palette name.
dark_palette_name	A string representation of the dark palette name.

Returns

A dictionary with two palettes.

generate_default_palettes()

```
generate_default_palettes (
              self )
```

Generates a default set of palettes from the extracted colors.

Parameters

self	The object pointer.

Returns

A dictionary of default color palettes.

generate_mood_palettes()

```
generate_mood_palettes (
              self,
              light_palette_name = 'light-mood',
              dark_palette_name = 'dark-mood' )
```

Generates two mood palettes, starting with the dominant color in the image.

Parameters

self	The object pointer.
light_palette_name	A string representation of the light palette name.
dark_palette_name	A string representation of the dark palette name.

Returns

A dictionary with two palettes.

generate_palettes()

```
generate_palettes (
              self,
              palette_templates = None )
```

Generates palettes based on a dictionary of palette templates.

Note

Palette templates follow a certain structure. Each palette template has a name (key) and a dictionary (value). For a more thorough explanation please refer to the Configuration File page on the PyPalEx GitHub's Wiki page: https://github.com/AlTimofeyev/pypalex/wiki/Configuration-File

Parameters

self	The object pointer.
palette_templates	A dictionary of palette template dictionaries.

Returns

A dictionary of color palettes.

get_closest_to_goldilocks()

```
get_closest_to_goldilocks (
              self,
              color_name1,
              color_name2,
              dom_color_name,
              color_type = '' )
```

get_color_code_index()

```
get_color_code_index (
              self,
              color_name )
```

Gets the color code index of a base color as it would appear in a color palette.

A color code is the NAME of a color as it appears in terminal / CLI color palettes (i.e. color0, color1, color2, ... , color15). A color code index is the index of said color where it would appear in a color palette.

Note

For more information about these ANSI escape codes, here are some sources: https://en.wikipedia.org/wiki/ANSI_escape_code#8-bit https://stackoverflow.com/questions/4842424/list-of-ansi-color-escape-sequences/33206814#33206814 https://stackoverflow.com/questions/45782766/color-python-output-given-rrggbb-hex-v

Parameters

self	The object pointer.
color_name	A string that represents the base name of a color.

Returns

Integer value that represents the index of a color code in a palette.

get_color_index()

```
get_color_index (
              self,
              color_name )
```

Gets the index of a base color as it would appear in a linear cyclical array.

Note

For example of colors on color a wheel being shown in linear cyclical format, please refer to the internal code comment block for this function.

Parameters

self	The object pointer.
color_name	A string that represents the base name of a color.

Returns

Integer value that represents the index of a base color in a cyclical array.

get_goldilocks_colorset()

```
get_goldilocks_colorset (
              self,
              color_name1,
              color_name2,
              dom_color_name )
```

Retrieves a list of 3 color types.

The color types are chosen by using a color pair and the name of a dominant color.

Parameters

self	The object pointer.
color_name1	A string that represents the base name of a color.
color_name2	A string that represents the base name of a color.
dom_color_name	A string that represents the base name of the dominant color in the image.

Returns

A list of 3 color types.

load()

```
load (
              self,
              absolute_image_path,
              image_name = None )
```

Loads the Extrator class with the provided image.

Parameters

self	The object pointer.
absolute_image_path	A string that represents the absolute path to an image.
image_name	A string that represents the name of the image or any name you want to provide with the current image being used.

organize_extracted_dictionary()

```
organize_extracted_dictionary (
              self )
```

Organizes the extracted colors dictionary.

The reorganization of the extracted colors' dictionary is done so that the (key, value) pairs appear in a specific order. This will be useful if the user wants to export the raw hierarchy of the data.

Parameters

self	The object pointer.

run()

```
run (
              self )
```

Main method for Extractor class.

Performs extraction of colors.

Parameters

self	The object pointer.

set_color_format()

```
set_color_format (
              self,
              new_color_format,
              colors_dict = None )
```

Sets the color format of the colors in the extracted dictionary.

There are 4 color formats to choose from: hsv, rgb, hex and ansi.

Note

There is a loss in precision when converting between color formats more than once. If you would like to convert color formats more than once while maintaining precision of the colors, please use an auxiliary dictionary to store copies of the original extracted colors before converting to a different format.

Parameters

self	The object pointer.
new_color_format	A string that represents the format each color should have (e.g. 'hsv', 'rgb', 'hex', 'ansi').
colors_dict	A dictionary of color names and their color values (optional parameter and can be a palette dictionary).

6.1.4 Member Data Documentation

base_color_dict

```
base_color_dict
```

A dictionary of 2D numpy arrays for each of the 6 base colors.

color_format

```
color_format
```

A string that represents the format each color should have (e.g.

'hsv', 'rgb', 'hex', 'ansi').

extracted_colors_dict

```
extracted_colors_dict
```

A dictionary of extracted colors in [h,s,v] format.

hsv_img_matrix_2d

`hsv_img_matrix_2d`

A 2D numpy array of pixels from an image in [h,s,v] format.

image_name

`image_name`

The name of the image file, without any extension (e.g.

.jpg, .png, etc.).

ratio_dict

`ratio_dict`

A dictionary that holds the ratio of base colors in an image and is used to identify the dominant color in an image.

The documentation for this class was generated from the following file:

- Extractor.py

7 File Documentation

7.1 __main__.py File Reference

Main script for PyPalEx.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.__main__

Functions

- main ()

  Main script function.
- handle_args ()

  Handles the arguments passed to PyPalEx.
- extract_color_palettes ()

  Handles color extraction from image(s).
- setup_argument_parser ()

  Sets up the argument parser for command line arguments.
- check_sources (filepaths, path=None)

  Checks each of the sources provided and removes any bad sources.
- check_path (path)

  Check the path to make sure it exists.
- handle_config ()

  Handle the PyPalEx configuration file settings.
- set_global_args (args)

  Sets the global variables using the arguments.
- preview_and_save (extractor, save_type, img_index)

  Shows a preview of and saves the extracted color palette(s).
- check_source (filepath)

  Checks to make sure the path leads to a file.

Variables

- str CONFIG_FILENAME = 'palex-config.yaml'

  Filename of the configuration file.
- list PROPER_IMAGES = [ ]

  List of real/existing image file path(s).
- list FILENAMES = [ ]

  List of image filenames (contain file extensions).
- list IMAGE_NAMES = [ ]

  List of image names.
- str OUTPUT_PATH = ''

  The path to the output directory where all exported files will be saved.
- str EXPORT_FILE_FORMAT = 'json'

  The format of the files to be exported (e.g.
- str EXPORT_COLOR_FORMAT = 'hex'

  The format in which the extracted colors will be exported (e.g.
- dict EXPORT_PALETTE_TEMPLATES = {}

  Dictionary of palette templates that can be used to organize extracted colors into palettes to export.
- dict PALETTE_COLOR_TYPES_CONTAINED = {}

  Dictionary of the color types that are contained within each palette template.
- bool SAVE_CHECK = False

  Flag to check if user wants to save extracted color palettes.
- bool SHOW_PREVIEW = False

  Flag to show a preview of extracted palettes.
- bool ADAPTIVE_PALETTE = False

  Flag to generate 2 adaptive color palettes.
- bool MOOD_PALETTE = False

Flag to generate 2 mood color palettes.

- bool SAVE_RAW = False

  Flag to save raw extracted colors.

- bool PASTEL_L = False

  Flag to convert light color type to pastel.

- bool PASTEL_N = False

  Flag to convert normal color type to pastel.

- bool PASTEL_D = False

  Flag to convert dark color type to pastel.

- str LIGHT_MOOD_PALETTE_NAME = 'light-mood'

  The palette name of the light-themed mood palette.

- str DARK_MOOD_PALETTE_NAME = 'dark-mood'

  The palette name of the dark-themed mood palette.

- str LIGHT_ADAPTIVE_PALETTE_NAME = 'goldilocks-light'

  The palette name of the light-themed adaptive palette.

- str DARK_ADAPTIVE_PALETTE_NAME = 'goldilocks-dark'

  The palette name of the dark-themed adaptive palette.

- dict VALID_COLOR_SET

  A set of valid color names used to check user-defined color palettes from the configuration file.

7.1.1 Detailed Description

Main script for PyPalEx.

Used to run from the Command Line.

7.1.2 Author(s)

- Created by Al Timofeyev on February 2, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on March 22, 2023.
- Modified by Al Timofeyev on March 26, 2023.
- Modified by Al Timofeyev on April 7, 2023.
- Modified by Al Timofeyev on June 10, 2024.
- Modified by Al Timofeyev on July 8, 2024.
- Modified by Al Timofeyev on December 15, 2024.

7.2 arg_messages.py File Reference

Archive of messages to display for arguments supplied by user.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.arg_messages

Functions

- bad_source_message ()

  Generates an error message if the sources provided were not images.
- bad_path_message ()

  Generates an error message if the directory provided is not a valid directory.
- no_args_help_message ()

  Generates a help message if no arguments were presented.

7.2.1 Detailed Description

Archive of messages to display for arguments supplied by user.

7.2.2 Author(s)

- Created by Al Timofeyev on March 3, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on July 8, 2024.

7.3 constants.py File Reference

A collection of constants for PyPalEx.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.constants

Variables

- list BLACK_RGB = [0, 0, 0]
- list WHITE_RGB = [255, 255, 255]
- list RED_RGB = [255, 0, 0]
- list ORANGE_RGB = [255, 153, 0]
- list YELLOW_RGB = [255, 213, 0]
- list CHARTREUSE_RGB = [191, 255, 0]
- list GREEN_RGB = [0, 255, 0]
- list SPRING_RGB = [0, 255, 149]
- list CYAN_RGB = [0, 255, 255]
- list AZURE_RGB = [0, 128, 255]
- list BLUE_RGB = [0, 0, 255]
- list VIOLET_RGB = [140, 0, 255]
- list MAGENTA_RGB = [255, 0, 255]
- list ROSE_RGB = [255, 0, 93]
- int BLACK_HEX = 0x000000
- int WHITE_HEX = 0xFFFFFF
- int RED_HEX = 0xFF0000
- int ORANGE_HEX = 0xFF9900
- int YELLOW_HEX = 0xFFD500
- int CHARTREUSE_HEX = 0xBFFF00
- int GREEN_HEX = 0x00FF00
- int SPRING_HEX = 0x00FF95
- int CYAN_HEX = 0x00FFFF
- int AZURE_HEX = 0x0080FF
- int BLUE_HEX = 0x0000FF
- int VIOLET_HEX = 0x8C00FF
- int MAGENTA_HEX = 0xFF00FF
- int ROSE_HEX = 0xFF005D
- int RED_HUE = 0
- int ORANGE_HUE = 36
- int YELLOW_HUE = 50
- int CHARTREUSE_HUE = 75
- int GREEN_HUE = 120
- int SPRING_HUE = 155
- int CYAN_HUE = 180
- int AZURE_HUE = 210
- int BLUE_HUE = 240
- int VIOLET_HUE = 273
- int MAGENTA_HUE = 300
- int ROSE_HUE = 338
- list RED_HUE_RANGE_MIN = [0, 20]
- list ORANGE_HUE_RANGE = [20, 43]
- list YELLOW_HUE_RANGE = [43, 64]
- list CHARTREUSE_HUE_RANGE = [64, 90]
- list GREEN_HUE_RANGE = [90, 145]
- list SPRING_HUE_RANGE = [145, 170]
- list CYAN_HUE_RANGE = [170, 195]
- list AZURE_HUE_RANGE = [195, 220]
- list BLUE_HUE_RANGE = [220, 255]
- list VIOLET_HUE_RANGE = [255, 290]
- list MAGENTA_HUE_RANGE = [290, 325]
- list ROSE_HUE_RANGE = [325, 350]
- list RED_HUE_RANGE_MAX = [350, 360]

- list BLACK_BRIGHTNESS_RANGE = [0.0, 35.0]
- list DARK_BRIGHTNESS_RANGE = [35.0, 55.0]
- list NORM_BRIGHTNESS_RANGE = [55.0, 80.0]
- list LIGHT_BRIGHTNESS_RANGE = [80.0, 100.0]
- list SATURATION_TOLERANCE_RANGE = [15.0, 20.0]
- list PASTEL_SATURATION_RANGE = [20.0, 55.0]
- list PASTEL_BRIGHTNESS_RANGE = [65.0, 95.0]

7.3.1 Detailed Description

A collection of constants for PyPalEx.

7.3.2 Author(s)

- Created by Al Timofeyev on February 2, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on May 31, 2024.
- Modified by Al Timofeyev on June 10, 2024.
- Modified by Al Timofeyev on October 12, 2024.

7.4 conversion_utils.py File Reference

Utilities for converting between RGB, HSV, HEX.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.conversion_utils

Functions

- hsv_to_hex (hsv_array)

  Convert HSV array [h,s,v] to HEX string '#ffffff'.
- hex_to_hsv (hex_str)

  Convert HEX string '#ffffff' to HSV array [h,s,v].
- hsv_to_ansi (hsv_array, background=False)

  Convert HSV array [h,s,v] to an ANSI color escape code string.
- ansi_to_hsv (ansi_string)

  Converts ANSI color escape code string tp HSV array [h,s,v].
- hex_to_ansi (hex_str, background=False)

  Convert HEX string '#ffffff' to an ANSI color escape code string.
- ansi_to_hex (ansi_string)

  Converts ANSI color escape code string to HEX string '#ffffff'.

- rgb_to_hsv (rgb_array)

  Converts RGB array [r,g,b] to HSV array [h,s,v].
- hsv_to_rgb (hsv_array)

  Convert HSV array [h,s,v] to RGB array [r,g,b].
- rgb_to_hex (rgb_array)

  Convert RGB array [r,g,b] to HEX string '#ffffff'.
- hex_to_rgb (hex_str)

  Convert HEX string '#ffffff' to RGB array [r,g,b].
- rgb_to_ansi (rgb_array, background=False)

  Convert RGB array [r,g,b] to an ANSI color escape code string.
- ansi_to_rgb (ansi_string)

  Converts ANSI color escape code string to an RGB array.

7.4.1 Detailed Description

Utilities for converting between RGB, HSV, HEX.

7.4.2 Author(s)

- Created by Al Timofeyev on February 2, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on April 5, 2023.
- Modified by Al Timofeyev on July 8, 2024.
- Modified by Al Timofeyev on October 12, 2024.

7.5 extraction_utils.py File Reference

Utilities for extracting colors from the image.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.extraction_utils

Functions

- extract_ratios (hsv_img_matrix_2d)

  Extracts the ratios of hues per pixel.
- construct_base_color_dictionary (hsv_img_matrix_2d)

  Constructs dictionary of base colors from an array of HSV pixel values.
- extract_colors (base_color_dict, ratios=None)

  Extracts dominant light, normal and dark colors from each of the base colors.
- check_missing_colors (base_color_dict, extracted_colors_dict)

  Checks for any missing colors in the base color dictionary and borrows them from the surrounding colors.
- generate_remaining_colors (extracted_colors_dict, ratios)

  Generate the remaining black and white, and background and foreground colors.
- extract_color_types (color_data)

  Extracts the dominant color types from a base color.
- set_missing_color (extracted_colors_dict, color_name)

  Sets a specified color in the extracted colors dictionary by borrowing from other extracted colors.
- get_dominant_hue (extracted_colors_dict, ratios)

  Calculates the dominant hue.
- generate_black_and_white (dominant_hue)

  Generates black and white color types using the dominant hue.
- generate_background_and_foreground (dominant_hue, complementary_hue)

  Generates the background and foreground colors.
- sort_by_sat_and_bright_value (hsv_base_color_matrix, color_name, ratios)

  Sorts the colors by their saturation and brightness values.
- extract_dominant_color (hsv_color_type_matrix)

  Extracts the dominant color from a color type.
- check_missing_color_types (light_color, norm_color, dark_color, black_color, achromatic_light, achromatic↩_norm, achromatic_dark, achromatic_black)

  Checks to make sure all the color types have been properly set.
- get_left_and_right_colors (origin_color_name, color_type='')

  Gets the color names of the colors that are to the left and right of the originating color.
- borrow_color (extracted_colors_dict, origin, borrow_left, borrow_right, color_type='')

  Borrows a color from one of the extracted color types of the base colors.
- get_dominant_color_name (ratios)

  Get the base name of the dominant color from a dictionary of ratios.
- calculate_centroid (hsv_color_type_matrix)

  Calculates the centroid for a color type.
- find_closest_to_centroid (hsv_color_type_matrix, centroid)

  Finds a color from a color type that is closest to the centroid.
- get_hue_shift_value (hue, shift_percentage)

  Gets the appropriate percentage to shift a hue value based on the hue provided.
- calculate_dist_between_2_colors (hsv_color1, hsv_color2)

  Calculates the distance between 2 HSV colors.

7.5.1 Detailed Description

Utilities for extracting colors from the image.

7.5.2 Author(s)

- Created by Al Timofeyev on February 10, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on March 22, 2023.
- Modified by Al Timofeyev on April 6, 2023.
- Modified by Al Timofeyev on May 31, 2024.
- Modified by Al Timofeyev on June 10, 2024.
- Modified by Al Timofeyev on July 8, 2024.
- Modified by Al Timofeyev on October 12, 2024.

7.6 Extractor.py File Reference

Extraction utility class for extracting colors from the image.

Classes

- class Extractor

  Extracts colors given a matrix of HSV values extracted from an image.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.Extractor

7.6.1 Detailed Description

Extraction utility class for extracting colors from the image.

7.6.2 Author(s)

- Created by Al Timofeyev on February 10, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on March 22, 2023.
- Modified by Al Timofeyev on April 5, 2023.
- Modified by Al Timofeyev on June 10, 2024.
- Modified by Al Timofeyev on July 8, 2024.
- Modified by Al Timofeyev on October 12, 2024.

7.7 file_utils.py File Reference

Utilities for file handling.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.file_utils

Functions

- generate_config_file (config_filename)

  Generates a configuration file.
- raw_dump (extracted_colors_dict, image_name, output_path, export_file_format, export_color_format)

  Saves the raw extracted colors into a file.
- save_palettes (palettes, image_name, output_path, export_file_format, export_color_format, palette_color↩ _types=None)

  Saves the color palettes of extracted colors.

7.7.1 Detailed Description

Utilities for file handling.

Note

Potential point for contributors to add different output saving options.

7.7.2 Author(s)

- Created by Al Timofeyev on April 5, 2023.
- Modified by Al Timofeyev on July 8, 2024.
- Modified by Al Timofeyev on October 12, 2024.

7.8 image_utils.py File Reference

Utilities for processing image and file handling.

Namespaces

- namespace pypalex

  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.image_utils

Functions

- process_image (image)
  Processes PIL Image object.
- rescale_image (image)
  Rescales image to a smaller sampling size while maintaining aspect ration.
- process_helper (rgb_matrix_2d)
  Helper function for multiprocessing conversion operations.

7.8.1 Detailed Description

Utilities for processing image and file handling.

7.8.2 Author(s)

- Created by Al Timofeyev on February 27, 2022.
- Modified by Al Timofeyev on April 21, 2022.
- Modified by Al Timofeyev on March 6, 2023.
- Modified by Al Timofeyev on April 5, 2023.
- Modified by Al Timofeyev on May 16, 2024.

7.9 print_utils.py File Reference

Utilities for printing preview to the screen.

Namespaces

- namespace pypalex
  Python Palette Extractor: extracts color palettes from images.
- namespace pypalex.print_utils

Functions

- print_raw_colors (extracted_colors_dict, color_format, pair_colors=False)
  Prints raw extracted colors to the Terminal.
- print_palette_preview (palettes, color_format)
  Prints the extracted colors, organized into color palettes, to the Terminal.
- print_default_palette_preview (extracted_colors_dict, color_format)
  Prints the extracted colors, organized into default color palettes, to the Terminal.
- print_template_palette_preview (extracted_colors_dict, palette_templates, color_format)
  Prints the extracted colors, organized with palette templates, to the Terminal.
- print_raw_color (color_name, extracted_color, rgb_color, ansi_color)
  Prints a raw color to the Terminal screen.
- get_rgb_colors (extracted_colors_dict, color_format)
  Constructs a dictionary of colors in RGB [r,g,b] format.

- get_ansi_color_codes (rgb_colors_dict)

  Constructs an ANSI escape code dictionary using a dictionary of colors in RGB [r,g,b] format.
- make_default_row (rgb_row_color, blank_row, border_type=None)

  Creates a string that represents a default row when printing palette previews.
- make_foreground_row (rbg_foreground_color, rbg_background_color)

  Creates a string that represents the foreground row when printing palette previews.
- make_panes (background_ansi_color, standard_ansi_colors, intense_ansi_colors)

  Creates a string that represents the 4 rows of panes when printing palette previews.
- make_panes_row (background_ansi_color, standard_ansi_colors, intense_ansi_colors, panes_section)

  Creates a string that represents a row of panes for printing palette previews.

7.9.1 Detailed Description

Utilities for printing preview to the screen.

Note

Potential point for contributors to add different printing options, maybe even a printing option that displays in a GUI.

7.9.2 Author(s)

- Created by Al Timofeyev on April 5, 2023.
- Modified by Al Timofeyev on July 8, 2024.
- Modified by Al Timofeyev on October 12, 2024.

Index